Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет Компьютерных Технологий и Управления

Кафедра Информатики и прикладной математики

**Системное программное обеспечение**

Лабораторная работа №2

Выполнил:

Хлопков Д. С.

Группа: Р3217

Преподаватель:

Зыков Анатолий Геннадьевич

Санкт-Петербург

2018 г.

**Листинг программы:**

1. #pragma warning(disable:4047)
2. #pragma warning(disable:4133)
3. #include <stdio.h>
4. #include <stdlib.h>
5. #include <string.h>
6. #include <stdbool.h>
7. #include <windows.h>
9. #define MAX\_BUFFER 1024
11. #define SORT\_FILE\_1 "bin1"
12. #define SORT\_FILE\_2 "bin2"
13. #define SORT\_FILE\_3 "bin3"
15. int check(int num){
16. if (num < 0)
17. {
18. num = num\*(-1);
19. }
20. return num;
21. };
23. int symbols(int Y)
24. {
25. int count = 0;
26. do {
27. Y = Y / 10;
28. count++;
29. } while (Y >= 1);
30. return count;
31. }
33. void Line(int symb1, int symb2, int symb3, int symb\_, int width) {
34. printf("%c", symb1);                           *// Г*
35. for (int j = 0; j<width; j++) printf("%c", symb\_); *// -*
36. printf("%c", symb2);                           *// T наоборот*
37. for (int j = 0; j<width; j++) printf("%c", symb\_); *// -*
38. printf("%c", symb2);                           *// T наоборот*
39. for (int j = 0; j<width; j++) printf("%c", symb\_); *// -*
40. printf("%c", symb2);                           *// T наоборот*
41. for (int j = 0; j<width; j++) printf("%c", symb\_); *// -*
42. printf("%c", symb3);                           *// L влево*
43. printf("**\n**");
45. }
47. void Line2(int palka, int space, int width) {
48. int j;
49. printf("%c", palka);
50. for (j = 0; j < width; j++) printf("%c", space);
51. printf("%c", palka);
52. for (j = 0; j < ((width - 4) / 2+1); j++) printf("%c", space);
53. printf("Heap");
54. for (j = 0; j < ((width - 4) / 2); j++) printf("%c", space);
55. printf("%c", palka);
56. for (j = 0; j < ((width - 7)/2); j++) printf("%c", space);
57. printf("Mapping");
58. for (j = 0; j < ((width - 7)/2); j++) printf("%c", space);
59. printf("%c", palka);
60. for (j = 0; j < ((width - 11)/2+1); j++) printf("%c", space);
61. printf("Basepoints");
62. for (j = 0; j < ((width - 11)/2); j++) printf("%c", space);
63. printf("%c**\n**", palka);
64. }
66. void Line3(int palka, int space, int width, int one, int two, int three, char\* method) {
67. int j;
68. printf("%c", palka);
69. for (j = 0; j < ((width - strlen(method)) / 2); j++) printf("%c", space);
70. printf("%s", method);
71. for (j = 0; j < ((width - strlen(method)) / 2 + strlen(method) % 2) - 1; j++) printf("%c", space);
72. printf("%c", palka);
73. for (j = 0; j < ((width - symbols(one) - 3) / 2 + symbols(one) % 2); j++) printf("%c", space);
74. printf("%d ms", one);
75. for (j = 0; j < ((width - symbols(one) - 3) / 2); j++) printf("%c", space);
76. printf("%c", palka);
77. for (j = 0; j < ((width - symbols(two) - 3)/2 + symbols(two) % 2); j++) printf("%c", space);
78. printf("%d ms", two);
79. for (j = 0; j < ((width - symbols(two) - 3)/2); j++) printf("%c", space);
80. printf("%c", palka);
81. for (j = 0; j < ((width - symbols(three) - 3)/2 + symbols(three) % 2); j++) printf("%c", space);
82. printf("%d ms", three);
83. for (j = 0; j < ((width - symbols(three) - 3)/2); j++) printf("%c", space);
84. printf("%c**\n**", palka);
85. }
87. void write\_table(int one, int two, int three) {
88. int width = 21;
89. Line(218, 194, 191, 196, width);
90. Line2(179, 32, width);
91. Line(195, 197, 180, 196, width);
92. Line3(179, 32, width, one, two, three, "Binary inserts ");
93. Line(192, 193, 217, 196, width);
94. }
95. *//сортировка бмнарными вставками*
96. void sort\_binary\_inserts(int\* arrayPtr, int size)
97. {
98. int x;
99. int left;
100. int right;
101. int mid;
102. for (int i = 1;  i < size; i++)
103. if (arrayPtr[i-1] > arrayPtr[i]){
104. x = arrayPtr[i];
105. left = 0;
106. right = i-1;
107. do {
108. mid = (left + right)/2;
109. if  (arrayPtr[mid] < x ) left = mid + 1;
110. else  right = mid - 1;
111. } while (left <= right);
112. for (int j = i - 1; j >= left; j--)
113. arrayPtr[j+1] = arrayPtr[j];
114. arrayPtr[left] = x;
115. }
116. }
118. int cmp(const void \* a, const void \* b) {
119. return (\*(int\*)a - \*(int\*)b);
120. }
122. void sort\_heaps(char\* filename) {
123. FILE\* file;
124. long int size;
125. int count;
126. LPVOID lpMem;
127. int\* buffer;
128. HANDLE heap;
129. int a;
131. file = fopen(filename, "rb");
132. fseek(file, 0L, SEEK\_END);
133. size = ftell(file);
134. fseek(file, 0L, SEEK\_SET);
135. count = size / sizeof(int);
136. heap = HeapCreate(0, 0x01000, 0);
137. lpMem = HeapAlloc(heap, 0, size);
138. buffer = (int\*)lpMem;
139. fread(buffer, sizeof(int), count, file);
140. fclose(file);
142. sort\_binary\_inserts(buffer, count);
144. HeapFree(heap, 0, lpMem);
145. HeapDestroy(heap);
146. }
148. void sort\_mapping(char\* filename) {
149. HANDLE hFile, hMap;
150. DWORD dwFileSize;
151. int\* memory;
153. hFile = CreateFile(filename, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL, OPEN\_EXISTING, 0, NULL);
154. hMap = CreateFileMapping(hFile, NULL, PAGE\_READWRITE, 0, 0, NULL);
155. dwFileSize = GetFileSize(hFile, NULL);
156. memory = (int\*)MapViewOfFile(hMap, FILE\_MAP\_WRITE, 0, 0, dwFileSize);
157. sort\_binary\_inserts(memory, dwFileSize/sizeof(int));
158. UnmapViewOfFile(memory);
159. CloseHandle(hMap);
160. CloseHandle(hFile);
161. }
163. void sort\_basepoints(char\* filename) {
164. int \*it, \*end;
165. HANDLE hFile, hMap;
166. DWORD dwFileSize;
167. char\* memory;
168. int\* data;
169. int i, count;
171. hFile = CreateFile(filename, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL, OPEN\_EXISTING, 0, NULL);
172. hMap = CreateFileMapping(hFile, NULL, PAGE\_READWRITE, 0, 0, NULL);
173. dwFileSize = GetFileSize(hFile, NULL);
174. count = dwFileSize / sizeof(int);
175. memory = (char\*)MapViewOfFile(hMap, FILE\_MAP\_WRITE, 0, 0, dwFileSize);
176. data = (int\*)malloc(dwFileSize);
178. i = 0;
179. it = (int\*) memory;
180. end = it + dwFileSize/sizeof(int);
181. for(it; it < end; it++)
182. data[i++] = \*it;

185. sort\_binary\_inserts(data, count);
187. i = 0;
188. it = (int\*)memory;
189. end = it + count;
190. for(it; it < end; it++)
191. (\*it) = data[i++];
193. UnmapViewOfFile(memory);
194. CloseHandle(hMap);
195. CloseHandle(hFile);
196. }
198. int main() {
199. SYSTEMTIME st, ft;
200. int one, two, three, four, five, six;
202. GetSystemTime(&st);
203. sort\_heaps(SORT\_FILE\_1);
204. GetSystemTime(&ft);
205. one = ft.wMilliseconds - st.wMilliseconds;
206. one = check(one);
208. GetSystemTime(&st);
209. sort\_mapping(SORT\_FILE\_2);
210. GetSystemTime(&ft);
211. two = ft.wMilliseconds - st.wMilliseconds;
212. two = check(two);
214. GetSystemTime(&st);
215. sort\_basepoints(SORT\_FILE\_3);
216. GetSystemTime(&ft);
217. three = ft.wMilliseconds - st.wMilliseconds;
218. three = check(three);
220. write\_table(one, two, three);
222. return 0;
223. }

**Результат работы программы:**

